

Ghost-in-the-Wireless: Energy Depletion Attack on ZigBee

Devu Manikantan Shila, Xianghui Cao, *Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, Zequ Yang, Yang Zhou, and Jiming Chen, *Senior Member, IEEE*

Abstract—ZigBee has been recently drawing a lot of attention as a promising solution for ubiquitous computing. The ZigBee devices are normally resource-limited, making the network susceptible to a variety of security threats. This paper presents a severe attack on ZigBee networks termed as *ghost*, which leverages the underlying vulnerabilities of the IEEE 802.15.4 security suites to deplete the energy of the devices. We manifest that the impact of *ghost* is severe as it can reduce the lifetime of devices from years to days and facilitate a variety of threats including denial of service and replay attacks. We highlight that merely deploying a standard suite of advanced security techniques does not necessarily guarantee improved security, but instead might be leveraged by adversaries to cause severe disruption in the network. We propose several recommendations on how to localize and withstand the *ghost* and other related attacks in ZigBee networks. Extensive simulations are provided to show the impact of the *ghost* and the performance of the proposed recommendations. Moreover, physical experiments also have been conducted and the observations confirm the severity of the impact by the *ghost* attack. We believe that the presented work will aid the researchers to improve the security of ZigBee further.

Index Terms—ZigBee; security; energy depletion attack; countermeasures; experiments

1 INTRODUCTION

DUE to its expandability, low cost, ease-of-use, and minimal maintenance, IEEE 802.15.4 based ZigBee, has been recently drawing a lot of attention to become the most prevalent solution for ubiquitous computing in everyday life. Since the origin of ZigBee, ZigBee alliance has essentially targeted their efforts on building a global wireless language for myriad of everyday devices such as light switches, thermostats, smart devices, remote controls as well as more complex sensor devices found abundantly in the health care, commercial building and industrial automation sectors [1]–[5].

Most of the applications over ZigBee are, however, security sensitive. For instance, it is envisioned that in a smart grid network, automated smart meters will exchange information about the energy consumption of homes to utility companies in a timely fashion. If such information is delivered in a plain-text manner, others could retrieve sensitive private information about the home residents such as their living habits and the time they are not at home. Malicious ones can also inject false energy use information to interrupt the billing system [6]. The IEEE 802.15.4 standard addresses the security requirements through a medium access control (MAC)

layer package, providing fundamental security services ranging from data confidentiality, data integrity to replay protection [7]. Despite that the standard provides these basic services, Sastry *et al.* outlined a number of security problems and pitfalls when using the IEEE 802.15.4 specification, especially pertaining to the initialization vector management, key management and integrity protection [8], and they also suggested several recommendations to improve the security posture of the specification. Zheng *et al.* presented more attacks on the physical and MAC sub-layers, including jamming, capture and tampering, exhaustion, collision and unfairness [9]. Besides research efforts, nowadays, off-the-shelf attack toolkits like KillerBee [10] are available that can be leveraged even by a novice adversary to explore and exploit the security of ZigBee networks. Using KillerBee and an IEEE 802.15.4 compatible radio interface, an adversary can carry out several attacks ranging from surreptitious eavesdropping to traffic injection with a little or no effort.

Markedly, people need to have a solid understanding of the security performance of ZigBee before positioning it as a major player in the market of ubiquitous computing. In this paper, our in-depth analysis of the ZigBee standard identified a potential flaw related to sending security headers in clear text. These security headers are treated as critical parameters in the specification to provide semantic security and replay protection. A key question is: *what might be the consequences if a malicious one masquerades as a trusted device by crafting the security headers?* IEEE 802.15.4 provides protection to integrity related attacks by including a cryptographically secure checksum (aka message integrity code (MIC)) with each message sent to a recipient. When an adversary crafts an invalid security header without knowing the key

- D. Shila is with United Technologies Research Center, Hartford, CT 06108 USA.
- X. Cao and Y. Cheng are with Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616, USA. Email: {xcao10,cheng}@iit.edu.
- Z. Yang, Y. Zhou and J. Chen are with Department of Control, Zhejiang University, Hangzhou 310027, China.

generating the MIC, although the integrity attack fails, *the recipient device in fact expends certain amount of energy receiving and processing those bogus messages*. In particular, it is shown that the security computing energy can be far from ignorable [11], [12]. If an intelligent adversary sends a number of such crafted messages to the victim device, a significant amount of energy will be used leading to battery depletion of the device.

There are studies on jamming attacks, MAC misbehavior, sleep deprivation via power attacks, and link layer exhaustion attacks in wireless networks [9], [13], [14]. Smart adversaries may attack the protocols in a deeper level, for instance Temporal Key Integrity Protocol (TKIP) MIC attack in IEEE 802.11 networks in which an attacker decodes the payload one byte at a time by using multiple replays and observing the response over the air on MIC failures [15], TCP SYN attack in which an attacker sends a chain of SYN requests to a victim system in an attempt to consume enough server resources and launch denial of service (DoS) attack [16] against public key cryptographic operations in sensor networks [17]. These attacks thwart the legitimate devices from using the medium and thereby are able to consume large amounts of victim devices' energy, referring to [18], [19] and the references therein. As opposed to these efforts, this paper demonstrates a novel and severe attack termed as *ghost-in-the-wireless* (aka *ghost*) on commercial ZigBee networks (with symmetric keys), in which a malicious one constructs bogus messages to lure the receiver to do superfluous security-related computations to intentionally deplete the energy of devices. The aftermath of the *ghost* attack is perilous as it will cut back the lifetime of devices from years to days (to be demonstrated by our simulation and experimental results) and further facilitate an adversary to execute a variety of after-depletion threats like denial of service, replay attack and loss of confidentiality.

In the literature, there also resource depletion attacks in many wireless networks. For example, adversaries can repeatedly send connection requests to exhaust the energy of implantable medical devices [20]. In ad hoc sensors networks, an adversary can drain the energy of sensors by purposely sending messages to construct artificial routing paths or introduce loops to the routing process of legitimate sensors [21]. However, how to execute such attacks on ZigBee is unknown yet. Moreover, the seriousness of *ghost* attack on commercial ZigBee networks, where devices that run on batteries have stringent power constraints, has not been explored.

In this paper, we exploit the ZigBee security headers to launch the *ghost* attack. We highlight the fact that merely deploying standard advanced security protocols does not necessarily ensure improved security, but instead might be leveraged by attackers to cause severe disruption in the network. Moreover, we provide qualitative analysis and quantitative simulations as well as experiments to demonstrate the severity of the *ghost* attack. We believe that our work in this paper will aid

researchers to further improve the security posture of energy-constrained wireless networks.

In summary, this paper has five-fold contributions:

- We propose a novel and severe attack on ZigBee networks, termed as *ghost*, which exploits the underlying vulnerabilities of the IEEE 802.15.4 security suites to cause an intentional energy failure of devices.
- We theoretically analyze the impact of the *ghost* attack on the victim node's lifetime, and develop an analytical model to quantify the impact of DoS attack (induced from *ghost*) on the throughput of a multi-hop chain network.
- We propose a three-phase algorithm to detect and localize the attacker based on network flow variations. We also propose several recommendations on how to withstand the *ghost* and its related attacks in ZigBee networks.
- Extensive computer simulation results are presented to demonstrate the impact of the *ghost* attack and the efficiency of the proposed solutions. Particularly, we show that the *ghost* attack can not only reduce the device lifetime from years to days by depleting energy, but also lead to the violation of the three basic principles of security, confidentiality, integrity and availability.
- We validate the effectiveness of *ghost* attack by conducting physical experiments on both a single-hop and a multi-hop ZigBee networks, where each node is equipped with CC2420 RF Transceiver that is compliant with the 2.4GHz IEEE 802.15.4 standard.

The rest of the paper is organized as follows. Section 2 reviews the IEEE 802.15.4 security architecture. Section 3 presents the *ghost* attack and some analysis. Section 4 discusses some other severe attacks stemming from the *ghost* attack while Section 5 proposes the corresponding countermeasures. Section 6 presents the simulation results to demonstrate the impact of *ghost* and *ghost*-related attacks and the proposed solutions. Section 7 presents the physical experiment results. Finally, Section 8 gives the conclusion remarks.

2 SECURITY ARCHITECTURE

In this section, we mainly pay our attention to the security services provided by the IEEE 802.15.4 MAC layer [7]. The standard particularly provides four basic security services for use by the higher layer applications: access control, data integrity, data confidentiality, and sequential freshness for replay protection, each of which is described briefly in the sequel. The MAC layer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. The higher layer (i.e., the application layer) indicates its choice of the security suite by setting the *security control field* in the *auxiliary security header* of the message, which identifies 8 candidate security levels ranging from 000 to 111, as shown in Table 1.

The security level configuration can be adjusted message by message. The absence of security parameters indicates “no security by default”. We urge the interested readers to refer the excellent write up in [8] for details about the security architecture of the standard.

TABLE 1: Security suites in IEEE 802.15.4.

Security Level/Id	Security Suite	Confidentiality	Integrity
000	None	✗	✗
001	AES-CBC-MAC-32	✗	✓
010	AES-CBC-MAC-64	✗	✓
011	AES-CBC-MAC-128	✗	✓
100	AES-CTR	✓	✗
101	AES-CCM-32	✓	✓
110	AES-CCM-64	✓	✓
111	AES-CCM-128	✓	✓

Access Control: The IEEE 802.15.4 MAC layer protocol prevents unauthorized devices from participating in the network by maintaining a list of valid devices, commonly known as access control list (ACL). For each incoming message, the receiving device checks the source address against the list of valid addresses in the table. If there is a match, the message is either accepted or forwarded to the next hop, otherwise it is dropped. Although such an access control mechanism can keep out the unauthorized parties from participating in the network, a number of issues emerge such as the spoofing attacks (data integrity issue) where an adversary masquerade as a valid user through crafting messages (e.g., source address) to bypass the ACL checks.

Data Integrity: The standard resolves data integrity issue by including a message integrity code, which is computed by applying a hash function over the message and pre-shared secret key (aka the symmetric key) [22]. The MIC tag along with the message is then sent to the receiver. Upon reception, the receiver can validate the integrity by checking whether the received MIC tag can be regenerated using the same hash function, the shared symmetric key, and the received message. If positive, the integrity is considered as maintained, i.e., both the message and the MIC tag were not modified; otherwise, the received message will be discarded. The standard provides data integrity services through AES-CBC-MAC and AES-CCM with three possible lengths of the MIC tag, i.e., 32 bits, 64 bits, or 128 bits.

Data Confidentiality: Underpinning the goal of confidentiality are the encryption schemes. Besides data encryption, the semantic security is needed to ensure that the attacker cannot learn even the partial information about the messages that have been encrypted. A common approach to realizing semantic security is to leverage a unique *nonce*, typically a counter or random value, for each invocation of the encryption algorithm. The main purpose of a nonce is to add discrepancy to the encryption process when there is little or no discrepancy in the set of messages. Since the receiver

should rely on the same nonce to decrypt the messages, nonces are typically sent in the same message with the encrypted data in plain text, without keeping it secret. *This work is concerned with the malicious ones manipulating those nonces sent in plain text.* In the sequel, we will show that how an adversary can leverage the plain-text nonce to launch the *ghost* attack. To provide semantic security, both algorithms of AES-CTR and AES-CCM use a 13-bytes nonce, which consists of an 8-bytes source address and a 5-bytes counter (that comprises the frame counter (4 bytes) and the security control field (1 byte) defined in the auxiliary security header).

Sequential Freshness for Replay Protection: Although data confidentiality and data integrity can prevent the network from a variety of known threats such as eavesdropping and spoofing, these schemes cannot protect the network from replay attacks. With the IEEE 802.15.4 specification, the sender usually assigns a monotonically increasing frame counter to each message and the receiver rejects those messages with smaller sequence numbers than it has already seen. The efficiency of this scheme clearly depends on the amount of time it will take the frame counters to roll over. To avoid roll over issues, a 32-bits counter is used in the IEEE 802.15.4. In addition to replay protection, the frame counters are considered as an important input to the construction of nonces for providing semantic security.

3 GHOST-IN-THE-WIRELESS

In this work, we consider a ZigBee network in which devices are statically deployed and communicate with one another to form a multi-hop wireless backbone. One or more devices serve as the coordinator (or gateways) and provide services for the entire ZigBee network. This work assumes that all the devices are collaborative, behave normally, and follow the algorithms correctly by sending the messages periodically to the coordinator. Notice that there are already works that talk about detecting the nodes that does not obey algorithms [23]. Similar to the standard, the cryptographic mechanism presented in this work is based on the symmetric-key cryptography and uses keys that are provided by higher layer processes. The establishment and maintenance of the keys are outside the scope of this work. As stated in the standard, we assume a secure implementation of the cryptographic operations, and secure and authentic storage of the keying material.

In such an environment, we assume the presence of single or multiple *ghost* attackers, equipped with compatible IEEE 802.15.4 radios. As opposed to legitimate devices, the attacker devices have no power or memory constraints and are assumed to span the attacking range over a large number of devices. We assume a three-phase attack model: (a) *pre-attack phase* — in this phase, the attacker learns about the network by surreptitiously eavesdropping, capturing and reverse engineering the messages; (b) *Attack phase* — the attacker leverages the

learned information to execute the *ghost* attack; and (c) *post-attack/depletion phase* — once the energy of devices are depleted, attacker in this phase executes several other attacks such as replay attack or confidentiality attack, to be discussed in Section 4.

3.1 AES Suites

ZigBee uses AES [24] based security suites to provide fundamental security services like confidentiality, integrity and replay protection. In this section, we will initially shed light on the working of each of these algorithms and then present the proposed *ghost* attack.

AES-CTR for Encryption. Messages in this mode are encrypted and decrypted by XORing with the key stream produced by the AES encrypting sequential counter block values. Let $O = O_1, O_2, \dots, O_n$ denote the output keystream block. To encrypt a payload with AES-CTR, the encrypter partitions the plaintext, P , into n 128-bit blocks P_1, P_2, \dots, P_n ; notice that if the last block is not a multiple of 128 bits, zeros are padded to it. Each P block is then XORed with a block of the key stream O to generate the corresponding ciphertext, $C = C_1, C_2, \dots, C_n$. To avoid reuse of the same output stream O , AES-CTR requires the sender (encrypter) to generate a unique key stream per-block per message. The decryption process is similar to encryption process.

AES-CBC-MAC for Authentication. A message is authenticated by splitting the input I into n 128 bit blocks, with necessary padding. Let $I = I_1, \dots, I_n$ and $O = O_1, \dots, O_n$ denote the input and its corresponding output block, where $O = CIPH(k)[I]$ and $CIPH(k)[I]$ is the invocation of AES algorithm on the input block using the secret key k . The CBC-MAC mode is then defined as: $O_1 = CIPH(k)[I_1]$, $O_2 = CIPH(k)[I_2 \oplus O_1]$, \dots , $O_n = CIPH(k)[I_n \oplus O_{n-1}]$ and the final block is the MIC.

AES-CCM for Encryption and Authentication. This mode consists of two steps: computing the MIC tag using CBC-MAC and encrypting the message concatenated with the MIC tag using CTR mode. Let $P = P_1, P_2, \dots, P_n$ and $O = O_0, O_1, O_2, \dots, O_n$ denote the plaintext and the output keystream blocks. Then the CCM mode is defined as follows: $P_1 \oplus O_1, P_2 \oplus O_2, \dots, P_n \oplus O_n || MIC \oplus O_0$.

3.2 Attack Phase

AES-CTR, AES-CBC-MAC and AES-CCM depend on the encryptor to generate a unique keystream per message (O) to provide semantic security. This task is accomplished by leveraging a 16-bytes unique counter constructed from the fields in the message intended for the destination. The 16-bytes counter consists of a 2-bytes static flags field, a 13-bytes nonce field (comprising the sender address, the frame counter and the security level field), and a 1-byte block counter that numbers the 16-bytes blocks within the message. In addition to semantic security, the frame counters are used by the

security suite to enable replay protection as well. When receiving a message, the recipient compares the frame counter seen in the incoming packet to the highest value stored in the ACL table. If the incoming packet has a larger counter value than the stored one, the message is accepted and the new counter is used to update the ACL table; otherwise, the message is rejected. With a 4-bytes frame counter, an adversary can carry out a replay attack only after 2^{32} frames, which is considered cryptographically secure in practice.

In the IEEE 802.15.4 protocol, those message fields used to construct the 16-bytes counter need to be communicated in plain text by the sender. Suppose that a malicious one injected messages into the network with increasing frame counters. According to the standard, the receiving device will accept it as the incoming message has a larger counter value than the last observed one. However, on decrypting the message, receiver will understand that the message is corrupted by computing the checksum or the MIC and finally, will drop it. Although the integrity attack fails, the recipient device in fact expends a valuable amount of energy accepting and processing those bogus messages. If an intelligent attacker sends a number of such crafted bogus messages to the victim device to lure the receiver to do the superfluous security-related computations, a significant amount of energy will be spent leading to battery depletion. We therefore term this attack as *ghost-in-the-wireless*. Although there are resource depletion attacks, we show by *ghost* how such an attack can be executed in ZigBee. Our simulation results show that through *ghost* attack, one can cut back the lifetime of devices from years to days.

3.3 Analysis

We present an analytical model to quantify the effect of a *ghost* attacker on a victim device (denoted as D_i). Consider that D_i works in a duty-cycling mode, which is a common working mode for low-cost networks such as wireless sensor networks, with duty-cycle equal to $\lambda = \frac{\tau}{T}$ where τ is the duration of an active period and T is the length of a cycle. Once entering an inactive period (aka sleep period), the device turns off its radio and changes CPU state (i.e., forces CPU into some low-power mode) if no incoming message is being received or decrypted; otherwise, it turns off its radio once the incoming message is finished receiving, and changes CPU state once the message is finished processing. The attacker sends bogus messages with crafted security headers to D_i at a high frequency (say ρ packets per unit time), while only those transmitted during D_i 's active period will be effective in depleting its energy. Assume that the attacker messages are of the same size.

For the victim device, denote T_{rx} as the time spent by its radio to receive a bogus message, which depends on message length and data rate. Denote $P_{rx}, P_{cpu}^a, P_{cpu}^i$ and P_{cpu}^o as the power required by its radio when in

receiving mode and by its CPU when in active, idle, and low-power modes, respectively. The energy consumption of a wireless device depends on the amount of energy expended on the following states: transmitting, receiving, computing, idle and sleeping states. Based on these states, the energy consumption of D_i is divided into three costs: communication cost (i.e., when it is receiving) E_{comm} , computation cost (i.e., when processing the data) E_{comp} , and passive cost (i.e., when it is not involved in communication, say, sleeping) $E_{passive}$.

The computation cost of the device depends on the amount of energy expended for cryptographic operations (including decryption and MIC verification). To compute it, initially, the cost of performing a decryption/verification in unit of ampere-cycle is computed by taking the product of the total number of clock cycles taken by the AES algorithm and the average current drawn by each CPU cycle. The total energy cost is then computed by dividing the ampere-cycles by the clock frequency in cycles/second of a processor and further multiplying the result with the processor's operating voltage. Thus the computation cost for the decryption and verification of one message is $T_{dec}P_{cpu}^a$, where T_{dec} denotes the time required for decryption and verification of a bogus message, which depends on the specific security suite used. Taking the CPU cost during the packet receiving period into account, we have

$$E_{comp} = n_p (T_{dec}P_{cpu}^a + T_{rx}P_{cpu}^i) \quad (1)$$

where n_p is the number of messages that will be processed during an active period. $n_p \approx \left\lceil \frac{\tau}{\max\{T_a, \frac{\tau}{\rho}\}} \right\rceil$, where $T_a \triangleq T_{dec} + T_{rx}$ is the total time to receive and process one bogus message.

As we are concerned about the amount of energy spent by the victim device D_i in receiving the bogus packets from the *ghost* attacker, the communication cost will be mainly determined by the receiving cost and is

$$E_{comm} \approx \begin{cases} n_p T_a P_{rx}, & \text{if } n_p T_a \geq \tau \\ \tau P_{rx}, & \text{otherwise} \end{cases} = \max\{n_p T_a, \tau\} P_{rx} \quad (2)$$

The passive cost includes sleep cost and idle cost between the completion time for processing a current message and the arriving time of the next message. Hence,

$$E_{passive} = \begin{cases} (\tau - n_p T_a) P_{cpu}^i + (T - \tau) P_{cpu}^o, & \text{if } n_p T_a < \tau \\ (T - n_p T_a) P_{cpu}^o, & \text{otherwise.} \end{cases} \quad (3)$$

Then, the energy consumption of the victim device receiving and processing a message from the *ghost* attacker is computed as $E_p = E_{comm} + E_{comp} + E_{passive}$.

3.3.1 Number of Messages Leading to Depletion

Let $E_{residual}$ be the amount of energy available to the device and $E_{threshold}$ be the threshold level below which the device fails to participate in the network, then we have the following condition to be true for the *ghost* attack to be successful:

$$E_{residual} - mE_p \leq E_{threshold} \quad (4)$$

where m is the number of bogus messages for victim device. From eqn.(4), it follows that the *ghost* attacker should send at least $m \geq \frac{E_{residual} - E_{threshold}}{E_p}$ packets to successfully deplete the energy.

3.3.2 Lifetime Reduction

Let E_0 be the initial energy of the victim device. If no attacker presents, its lifetime is $L_0 = \frac{E_0}{E_{p,0}}$ where

$$E_{p,0} = \tau(P_{rx} + P_{cpu}^i), \quad (5)$$

if we neglect the sleep cost. If $n_p T_a \geq \tau$, based on the above analysis, we have $E_p = n_p(T_{dec}P_{cpu}^a + T_{rx}P_{cpu}^i) + n_p T_a P_{rx}$. Thus, the ratio of L_0 over the victim device's lifetime under *ghost* attack becomes

$$\begin{aligned} \frac{L_0}{L} &= n_p T_a \frac{\frac{T_{dec}}{T_a} P_{cpu}^a + \frac{T_{rx}}{T_a} P_{cpu}^i + P_{rx}}{\tau(P_{rx} + P_{cpu}^i)} \\ &= \frac{n_p T_a}{\tau} \left(1 + \frac{T_{dec}}{T_a} \frac{P_{cpu}^a - P_{cpu}^i}{P_{rx} + P_{cpu}^i} \right) > \frac{n_p T_a}{\tau} \end{aligned} \quad (6)$$

That is $\frac{L}{L_0} < \frac{\tau}{n_p T_a}$. Similarly, if $n_p T_a < \tau$,

$$\frac{L_0}{L} = 1 + \frac{n_p T_{dec}}{\tau} \frac{P_{cpu}^a - P_{cpu}^i}{P_{rx} + P_{cpu}^i} \quad (7)$$

Above two equations clearly show that the lifetime of the victim device is reduced to some extent, and the amount of reduction can be significantly large if $n_p T_a$ is much larger than τ .

4 OTHER ATTACKS DUE TO GHOST

In this section, we will demonstrate a number of severe attacks stemming from *ghost* attack and propose countermeasures to mitigate their impact.

4.1 Denial of Service (DoS)

DoS can be easily executed by a *ghost* attacker in three ways. (1) *DoS due to high computational load on the device*. In a *ghost* attack, the adversary sends a number of bogus messages to quickly deplete the energy of the victim device and thereby, suspend the availability of the services. It turns out that if the network has some traffic abnormality detection schemes in place, sending such numerous messages in short period of time can be easily caught. To escape from the detection, for instance, *ghost* attacker(s) can send messages either at different times or at different addresses to a subset of victim devices in its range. (2) *DoS due to MAC misbehavior*: The

IEEE 802.15.4 utilizes distributed contention resolution mechanisms such as CSMA/CA for sharing the wireless channel. Under CSMA/CA protocol, only one transmission can happen at any given point in time in a given area and therefore, to achieve this CSMA/CA requires devices to sense the channel for idleness before it can transmit. In such an environment, if a *ghost* attacker continuously sends the traffic to the victim host, all devices within the interference region will be deprived of channel access and services. Moreover, each device has to spend a significant amount of time sensing and waiting to get access to channel, which again will lead to energy depletion. (3) *DoS with a post-depletion replay attack*: Such an attack is to be discussed in Section 4.2.1.

4.1.1 Analysis

To illustrate the effect of DoS attack on the network throughput performance, we conduct mathematical analysis based on a simple multi-hop network as shown in Fig. 1. All the nodes apply the CSMA/CA protocol for channel access. Assume each node randomly generates packets (of the same length L) at the same rate λ packets/slot, where a slot is the backoff slot in IEEE 802.15.4 standard. For ease of analysis, we use the following CSMA/CA parameters: the minimum and maximum backoff exponents are the same as $macBE$, $CW=1$, i.e., a node will start to transmit packet if it finds the channel is idle in previous slot. The nodes send packets to the gateway through the paths shown in this figure. A node will switch to receiving mode only when its MAC buffer is empty. The gateway node keeps in receiving mode and listening packets from others. The *ghost* attacker sends bogus packets at a constant rate p_{att} independent of the channel state.

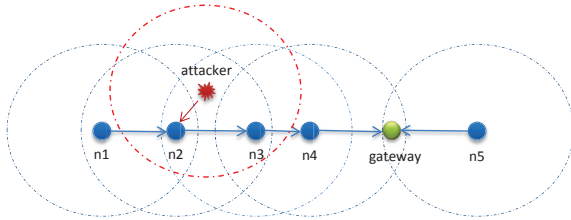


Fig. 1: A multi-hop network formed by 5 ZigBee nodes and a gateway, where node 2 and 3 are in the interference range of a *ghost* attacker.

For each node, define τ_i , α_i and ρ_i as the probabilities that this node conducts a clear channel assessment (CCA), the channel is idle when it conducts a CCA, and that there is at least one packet in its MAC buffer, respectively in a slot. Conditioned on that its buffer is nonempty, a node will transmit a packet with probability p_i and the packet will be successfully received (without collision) by its next-hop node with probability p_i^s . We extend the model in [25] for single-hop networks to the multi-hop network shown in Fig. 1. Since the modeling approach is the same for each node, below we shall focus on node 2. Since the backoff exponent is constant,

according to [25],

$$\tau_2 = \frac{1}{\bar{b} + 1 + L\alpha_2}, \quad (8)$$

where $\bar{b} \triangleq \frac{1}{2}(2^{macBE} - 1)$ is the average backoff length. When this node conducts a CCA, the channel is idle only when none of its neighbors (including the attacker) is transmitting (i.e., none of them starts transmitting in either of the previous L slots). Therefore,

$$\alpha_2 = 1 - L[1 - (1 - p_{att})(1 - \rho_1 p_1)(1 - \rho_3 p_3)]. \quad (9)$$

By definition,

$$p_2 = \tau_2 \alpha_2, \quad \rho_2 = \min \left\{ \frac{\lambda + \rho_1 p_1 p_1^s}{p_2}, 1 \right\}. \quad (10)$$

where $\rho_1 p_1 p_1^s$ is the rate of packets being received from node 1. When node 2 transmits a packet, the packet will be successfully received by node 3 only if: 1) neither node 1 nor the attacker starts to transmit packet simultaneously as node 2, 2) node 4 does not transmit packet in any slot during the transmission by node 2, and 3) node 3 is in receiving mode. Therefore,

$$p_2^s = (1 - \rho_1 p_1)(1 - p_{att})(1 - L\rho_4 p_4)(1 - \rho_3). \quad (11)$$

Based on the same approach, the performance of the other nodes can be modeled. Then, the throughput of each node can be calculated.

$$\begin{cases} S_1 &= \rho_1 p_1 p_1^s p_2^s p_3^s p_4^s, \\ S_2 &= \frac{\lambda}{\lambda + \rho_1 p_1 p_1^s} \rho_2 p_2 p_2^s p_3^s p_4^s, \\ S_3 &= \frac{\lambda}{\lambda + \rho_2 p_2 p_2^s} \rho_3 p_3 p_3^s p_4^s, \\ S_4 &= \frac{\lambda}{\lambda + \rho_3 p_3 p_3^s} \rho_4 p_4 p_4^s, \\ S_5 &= \rho_5 p_5 p_5^s \end{cases} \quad (12)$$

We name the scenario shown in Fig. 1 as case 1. For comparison, another case (case 2) is also considered in which the attacker's interference range only covers node 3. Numerical results are shown in Fig. 1, where $L = 3$, $macBE = 3$, $\lambda = 0.02$. In both cases, due to increase of attack intensity, nodes that are within the interference range of the *ghost* have fewer opportunities to send out packets, and their throughput drops quickly. Since packets from node 1 transverse the interfered area, its throughput drops even quicker than that of the nodes within the area. On the other hand, due to less interference from node 3, the throughput of node 4 and 5 whose packet routes do not cross the interfered area may gain some improvement. From Fig. 2(d), we observe that, even node 2 is a relay of node 1, they experience similar throughput variation when node 4, a common relay of them, is under attack. We take advantage of such observation to develop an attacker localization method in Section 5.1.

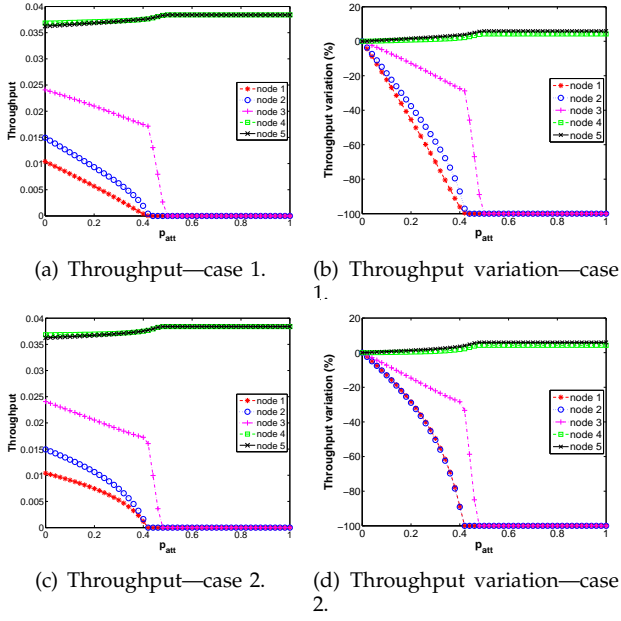


Fig. 2: Network performance under various attack rate. $p_{att} = 0$ means the scenario without *ghost*.

4.2 Post-Depletion Attacks

What happens when the device encounters an energy depletion? If no specific controls are taken, the device will emerge with a cleared ACL table on reboot and consequently, all the nonces or the frame counters will be reset to the initial value 0. Upon such resetting, we see the plausibility of two attacks:

4.2.1 Replay Attack due to Frame Counter Reuse

Replay attack can cause severe problems. For example, medical devices such as fitbits, pacemakers and insulin pumps which replay old messages can lead to erroneous outputs and dangerous events. Below we show that a *ghost* can launch replay attack. Consider a *ghost* attacker who captures a certain number of legitimate packets in the beginning (pre-attack phase) and starts depleting the energy of the devices through bogus messages. When the counters are reset to 0 upon restart, the attacker can start replaying those messages. The replay attack can lead to serious results in two aspects. (1) The attack can have the receiver consume obsolete messages, leading to unexpected operations. (2) The replay attack can result in the denial of service. Specifically, if the attacker replays messages with frame counters larger than the current legitimate value maintained in the ACL table, the ACL table will then be updated by such replayed messages. Consequently, in future, when legitimate devices send messages with the actual frame counter, the message will be rejected as it carries a counter value less than what is currently stored in the ACL of the victim device (which was updated by the replayed messages).

We would like to emphasize that the *ghost* attack significantly facilitates the chance of launching a replay attack. From the IEEE 802.15.4 specification, it follows

that by leveraging a 4-bytes counter, one can execute a replay attack only after 2^{32} frames; nonetheless, in this work, we manifest that by executing the *ghost* attack, an adversary can use a number of messages which is far less than 2^{32} to deplete the energy of a victim device. Upon the restart of this victim device, the attacker can launch the relay attack.

4.2.2 Loss of Confidentiality due to Nonce Reuse

According to the standard, messages destined for the next hop devices are encrypted by XORing the plaintext message P with the unique keystream O constructed from the unique nonces, static flags and the pre-shared key. Take the AES-CTR scheme for instance. To encrypt a payload, the encrypter partitions the plaintext, P , into n 128-bit blocks P_1, P_2, \dots, P_n ; if the last block is not a multiple of 128 bits, zeros are padded to it. Each P block is then XORed with a block of the key stream O to generate the corresponding ciphertext, $C = C_1, C_2, \dots, C_n = P \oplus O$. If the nonces are reused, then the keystream will repeat for the subsequent messages. Thus, for a different plain text P' , the output of the encryptor will be $C' = P' \oplus O$. Assume that the attacker has captured the ciphertexts C and C' , then a simple XORing of the two ciphertexts will lead to $C' \oplus C = (P' \oplus O) \oplus (P \oplus O) = P' \oplus P$; such a simple XOR of plain texts can trivially be broken using statistical analysis and the subsequent plaintext messages can be deciphered without the knowledge of pre-shared key, if the content of any one message can be guessed.

5 COUNTERMEASURES

In this section, we first consider the problem of *ghost* attacker localization. Next, we propose add-on mechanisms to current IEEE 802.15.4 standard to enhance network security against *ghost*.

5.1 Attacker Localization

Although the specific victim node is able to know the existence of a *ghost* attacker by checking whether the number of received bogus messages exceeds a tolerable level, it, under intense attack, may not be able to timely inform others since the channel could be unavailable (occupied by the *ghost*) and its CPU may be also busy in processing the bogus messages. Whereas, owing to the induced DoS impact on the network throughput (as demonstrated in Fig. 7 in our simulations), other nodes can still be able to detect and localize the attacker by analyzing the network flow changes. Consider the typical cluster based topologies in ZigBee networks. In a cluster, we let the cluster head (CH), the confluence of the flows from the other nodes, to carry out the analysis. As a first attempt to localize *ghost* attacker in such a ZigBee network, a three-phase method is proposed as follows. It is difficult, if not impossible, for the CH to determine the number of *ghosts* if they collocate or are very close by analyzing their DoS impact. However,

knowing the existence of at least one *ghost* at a location already satisfies our purpose.

Phase 1: Identify suspected victim nodes. The CH applies a moving analysis window to record the throughput of each node¹ in the corresponding cluster, and calculates the percentage of throughput variation (denoted as ΔS_i for node i) in real time. Suspected nodes are identified by checking the variation along each path. The observation mentioned in Section 3.3 is utilized to exclude those nodes outside of the attacker's interference area but their packet routes traverse that area. This exclusion operation is controlled by a threshold δ' as shown in Algorithm 1. We denote each path as $P_l = (l_1, l_2, \dots, l_m)$ with l_1 as the source node and l_m (i.e., the head) as the destination node. δ is used to eliminate throughput variance in normal condition.

Algorithm 1: Identify suspected victim nodes.

```

 $\mathcal{A} \leftarrow \emptyset$ : set of suspected victim nodes;
for all paths  $P$  do
  if  $\Delta S_{l_1} > -\delta$  then
    continue;
  end
  for  $i \leftarrow 1, \dots, m-1$  do
    if  $\Delta S_{l_i} < -\delta$  then
      if  $i+1 = m$  or  $|\Delta S_{l_i} - \Delta S_{l_{i+1}}| > \delta'$  then
        add  $l_i$  into  $\mathcal{A}$ ;
        if  $|\Delta S_{l_i} - \Delta S_{l_{i-1}}| < \delta'$  then
          add  $l_{i-1}$  into  $\mathcal{A}$ ;
        end
      end
    end
  end
end

```

Phase 2: Group suspected victim nodes. We consider a general case that neither the number nor interference ranges of the *ghosts* is known to the CH. Two suspected nodes can be considered under the interference of the same attacker and included in the same group if there is a circle that covers both of them but does not cover any other non-suspected nodes. Otherwise, they are considered to be attacked by different attackers and included in different groups. Since an attacker may want to span its attacking range over a large number of nodes, small clusters will not be considered for attacker localization.

Phase 3: Calculate the *Ghost* location. For each group, suppose there are s suspected nodes and the throughput of each node decreases by ΔS_i percent. Then, the attacker's location is estimated by the following weighted sum method.

$$L_{ghost} = \sum_{i=1}^s \frac{\Delta S_i}{\sum_{k=1}^s \Delta S_k} L_i, \quad (13)$$

1. For dense deployment, to save computation cost in analysis, the CH can select a few paths with the nodes scattering over the area as uniform as possible for analysis.

where L_i is the location of each suspected victim node.

5.2 Add-on Mechanisms

We propose to add techniques such as *blacklisting* to current security services provided by standard, in which each device will maintain a list of devices that are misbehaving. In the case of *ghost* attack, if the victim device observes a certain number of messages with bogus security headers, it will add the device to the blacklist and inform the network or the operator about the attack. One plausible issue with this approach is the *badmouthing* attack [23] in which an attacker sends bogus messages from different addresses and causes the victim device to blacklist all its surrounding devices, leading to a temporary disruption or denial of service.

Another approach is to add a second layer of *challenge-response scheme* in which the device, after observing a certain number of bogus messages from a specific address or when the energy is restored and the communications are re-initiated, will challenge the attacker with a random number. The solution requires the attacker to include the response to the challenge in the next message for the device. If the attacker is able to respond to it correctly, it will continue its operation, otherwise, it will inform the network or the operator about the attack. Notice that to include correct response in the message, the attacker needs to know the secret key which is available only to the legitimate parties and is securely stored in the device. On restoration of energy, we also require, as part of the challenge-response scheme, to establish new keys so that they don't reuse the same nonce twice with the same key. Another solution is to add a timestamp to the protocol and after a reboot, each device is required to update its timestamp by communicating with the controller. There are also works that suggest storing the counter values in non-volatile or flash memory so that even if the energy is lost, the state of the device can be restored. Nevertheless, storing and retrieving values from flash memory is slow and energy inefficient, specifically for energy constrained devices [8].

6 SIMULATIONS

In this section, we present extensive simulation results to demonstrate the impact of the proposed *ghost* attack. We develop our simulation codes within the NS-3 environment [26]. We consider a ZigBee network with a single coordinator serving as the gateway, n legitimate nodes and a *ghost* attacker which injects messages with bogus security headers to drain victim nodes' energy. Each ZigBee node is equipped with an 8 MHz processor. All the nodes apply the non-beacon mode of the IEEE 802.15.4 CSMA/CA protocol for channel access. We adopt the energy consumption model presented in [27]. Specifically, the current drain by each node for its CPU being in active, idle and power-save states are 8.0 mA, 3.2 mA and 110 μA , respectively; the current drain by each node's transceiver for receiving and transmitting

are 7.0 *mA* and 8.5 *mA*, respectively; and the transmission power of each node is 0 dB. We also adopt an accurate Li-Ion battery model (which is provided in NS-3 based on the models proposed in [28], [29]) for each node with the initial capacity set as 2.45 Ah to simulate the energy drain activities. We run the simulation program on a 1.85 GHz PC. Based on the number of instructions, the CPU computation time of the PC is mapped to that of the 8 MHz processors to approximate the computational energy consumption involved in the ZigBee security protocol.

6.1 Energy Expenditure for Security

We first compare the energy expenditures in encryption and decryption for the security suites listed in Table 1. To avoid the impact from other factors such as interference and routing, we simulate a two-node scenario in this experiment, i.e., a ghost attacker sends packets to a victim node. The victim node runs in a duty-cycling mode with fixed duty cycle at 1%, a typical value for sensor networks [30]. Specifically, the victim node stays in active state for channel listening for 1 ms in every 0.1 second. If there are packets arriving during the active period, the node will process the packets and then switch to sleep if there is still leftover time within the current 0.1-second cycle. If there is no packet received, the node will directly switch to sleep state upon expiration of the active period. Such a mechanism would ensure that the victim device could function properly over one year. To demonstrate the energy-depletion attack, the *ghost* attacker sends crafted messages to this victim node every 0.1 second within its active period.

Fig. 3 reports the average computational energy cost and time versus the MAC payload size of the bogus packet for different security suites, where the averages are taken over 25000 bogus packets sent by the attacker. Generally, both encryption and decryption energy costs grow as the payload size increases. Moreover, our simulation results in Fig. 3(a) and 3(b) demonstrate that the ascending order of the security suites in terms of the average per-packet energy cost (for both encryption and decryption) is as AES-CTR, AES-CBC-MAC, and AES-CCM. This order is reasonable because more data bytes are involved in the XORing operations when the protocol changes from AES-CTR to AES-CCM.

As both the encryption process and the decryption process pad additional 0's to the secured payload in order to partition input data into blocks of 128 bits (16 bytes), the amount of energy expenditure may remain the same for some payload sizes, which is shown in Fig. 3(b). For instance, with AES-CCM-32, a secured packet with MAC payload of 20 bytes will be padded by 10 bytes of 0's so that the resulting string, together with a 2-byte length indicator, has length (in bytes) divisible by 16. However, a MAC payload data of 30 bytes does not need extra 0's. As a result, the two secured packets with different MAC payload sizes consume almost the same

computational energy. It is also interesting to observe that, for some MAC payload lengths (e.g., 80 bytes in the figure), the average energy costs for decrypting a secured packet under the AES-CCM and AES-CBC-MAC security suites are quite close. In fact, after padding with necessary 0's, the total numbers of bytes (including secured data and authentication fields) that are input to the XORing operations are very close to each other in these two security suites.

The processing time by the victim node's CPU for decrypting and verifying a secured packet is much longer than that by its transceiver for receiving it. For instance, as shown in Fig. 3(c), with AES-CCM-128, the computation time for decrypting and verifying a packet with 60-byte payload is about 35 ms, while the receiving time is only about 3 ms. As shown in Fig. 3(d), over 88% of the victim device's energy is spent by its CPU for decrypting crafted packets, which clearly demonstrates the significance of the *ghost* attack. We can also observe that AES-CCM-32 introduces the highest CPU cost ratio. Note that the computation cost ratios in all situations slightly decrease when the MAC payload size increases. The reason is that the payload size does not significantly impact the computational energy cost, while it is linearly related to the transceiver's energy consumption in receiving.

6.2 Node Lifetime under the *Ghost* Attack

The *ghost* attack can significantly drain the energy of a victim node. For example, we can have a worst-case estimation according to the configuration given in Section 6.1. In a network where the active period in each 0.1-second cycle is around 1 ms, the ghost attacker sending packet with a payload size of 60 bytes can lead to a busy period around 35 ms.

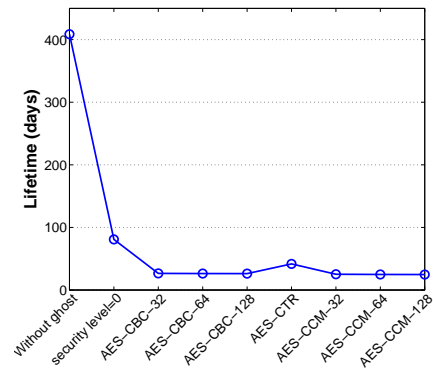


Fig. 4: Amount of time (in days) needed to deplete the energy, where the attacker sends bogus packets every 0.1 second.

Fig. 4 presents the lifetimes of the victim node in different scenarios including the case without attack and the cases with attacks (operated under the 8 security suites, respectively). The attack is according to the configuration in Section 6.1 with a MAC payload of 60 bytes.

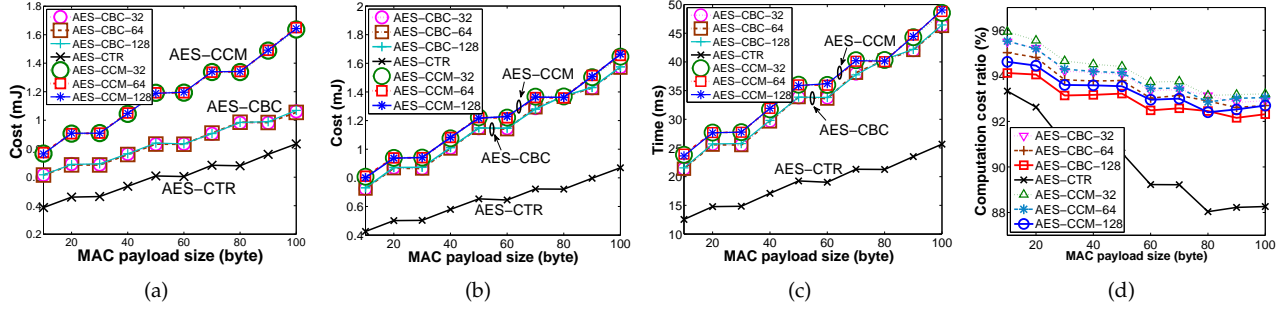


Fig. 3: Energy and time costs. (a) and (b): average energy consumption for encrypting a packet and decrypting a secured packet, respectively. (c): average time (ms) for decrypting and verifying a secured packet. (d): percentage of CPU computational cost in receiving (including decrypting) a secured packet.

The results in Fig. 4 clearly show that the ghost attack significantly shorten the lifetime from over one year to days. Specifically, the lifetime reduces to around 10.2%, 6.5% and 6.1% of the baseline value (i.e., without *ghost* attack) under AES-CTR, all the AES-CBC-MAC and all the AES-CCM security suites, respectively. On the other hand, based on the analysis presented in Section 3.3, we can easily calculate the lifetime ratio $\frac{L}{L_0}$ by using the computing time shown in Fig. 3(c) and taking into account the sleep cost. The calculated results are 10.9%, 6.8% and 6.5% for AES-CTR, all the AES-CBC-MAC and all the AES-CCM security suites, respectively, which well match the simulation results.

6.3 Denial of Service Attack

The following simulations are conducted on a network with $n = 38$ legitimate nodes randomly deployed inside a $100 \times 100 \text{ m}^2$ area, as shown in Fig. 5. The gateway node is located at the center of this area. All the nodes (including the attacker node) are assumed to have the same communication range (30m) and the same interference range (40m), respectively. All the legitimate nodes periodically report data (1 packet/s) to the gateway through multi-hop routes, where we adopt the shortest path routing. The attacker node select node 1 as the victim node and sends bogus packets to it, where the inter-packet sending intervals obey Poisson distribution with 20 ms as the mean. Here we interpret the attack as a DoS attack, according to the discussion in Section 4.1. Simulation results with AES-CTR are reported in the following.

Fig. 7(b) plots the throughput variation (i.e., the percentage of throughput that is increased/decreased due to the attack) for each node. Note that the red and blue bars mean positive and negative values, respectively. Based on this figure, the DoS attack by the *ghost* attacker will change the distribution of the network traffics, and the effects are two-fold. Since the attacker causes node 1 to spend most time on receiving and decrypting crafted packets, the spare capacity that node 1 can provide to others for relaying and also transmitting packets of its own significantly drops. Therefore, nodes, such as node

28, that needs node 1 to relay packets experiences a much higher packet loss rate in the presence of the attack. Although other nodes that receive a crafted packet will directly discard it due to unmatched destination address, there is still bandwidth and energy waste due to channel sensing, packet receiving, and collisions due to the hidden terminal impact [31]. Thus, the throughput of the attacker's neighboring nodes (e.g., node 8, 17 and 32) reduces greatly as shown in Fig. 7(b). Therefore, the attacker can perform denial of service attack at all its neighboring nodes by just targeting at one of them. On the other hand, the presence of the attacker may be beneficial to some other nodes, e.g., node 2, 10, 27, 31 and 37 as shown in Fig. 7(b), which are located far away. For those nodes, because the traffics of some hidden-terminal nodes locating within the attacker's interference range are partly suppressed by the attacker, their successful packet deliver ratios increase in turn.

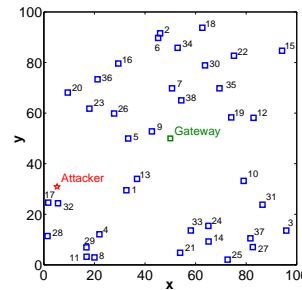


Fig. 5: Network topology

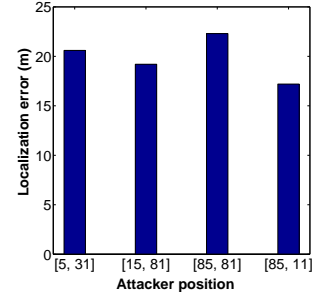
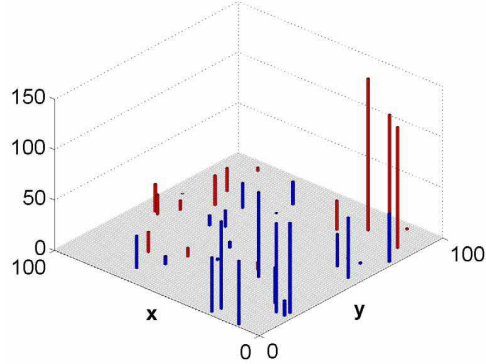
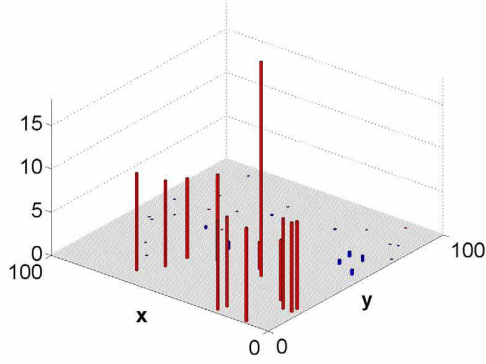


Fig. 6: Localization error.

Fig. 7(a) plots the 3D view of the variation in energy drain speed. We can see that, under the DoS attack, the energy drain speed is accelerated for most of the legitimate nodes. Therefore, the impact of the *ghost* attack can be propagated in a multi-hop network such that it can drain the energy of a larger amount of nodes more than that of its neighbors.

6.4 Effectiveness of Countermeasures

The performance of the proposed attacker localization method is demonstrated in Fig. 6 where the attacker is placed at four different positions. We can see that



(b) Throughput variation (%).

Fig. 7: Impact of DoS attack.

the localization error is around 20m, which is smaller than the communication range of each legitimate node. For denser networks, a lower localization error may be achieved since the positions of more interfered nodes can be utilized for calculating the attacker's position.

As shown in Fig. 8(a), the intrusion of the attacker significantly reduces the victim node's traffic, which is then recovered by blacklisting the attacker so that all future crafted packets are directly discarded by the victim node at the MAC layer. In terms of the energy drain speed, similar trends can be observed in Fig. 8(b). However, the drain speed cannot be fully recovered because the victim node's physical layer energy expenditure for receiving crafted packets is inevitable. In Section 5, we also proposed the challenge-response scheme to address both DoS and replay attacks. The traces of the victim nodes' traffic and energy drain rates exhibit similar patterns as shown in Fig. 8(a) and 8(b), and hence are omitted due to the page limit.

7 EXPERIMENTS

To further validate the effect of *ghost*, we conduct physical experiments based on the ZigBee nodes shown in Fig. 9. Each node has an ATmega128L processor operating with 8MHz frequency, 128K Bytes In-System programmable flash, and a CC2420 RF transceiver compliant with the 2.4GHz IEEE 802.15.4 standard. We develop C programs to activate the embedded security suites of IEEE 802.15.4 and control the security level.

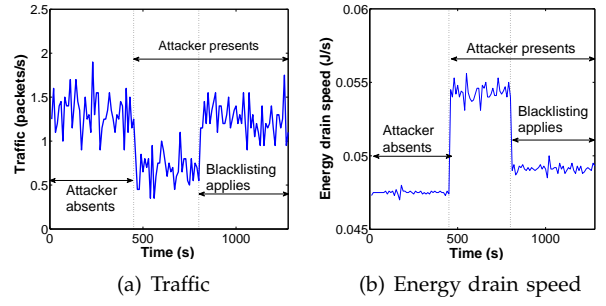


Fig. 8: The effect of blacklisting on the victim node's traffic (a) and energy drain speed (b).

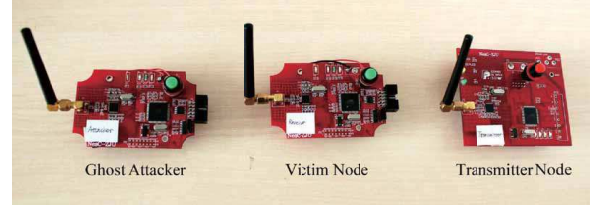


Fig. 9: ZigBee nodes in experiments.

By utilizing an ETCR6000 AC/DC Clamp Leaker with the sampling rate of 2 times per second, the working current under different modes of the victim node is measured and the mean values are shown in Table 2. In the idle state, i.e., MCU is in idle state and the radio is turned off, node's energy is consumed by the basic node operations including the glowing of LED light. When we maintain the MCU executing instructions (e.g., keeping on adding) incessantly, the energy consumption is raised by 7mA. Furthermore, when we turn on the radio, the node's working current is remarkably raised. Specifically, if the security suite is implemented into the procedure of transmission, the current reaches around 40mA; otherwise it reaches around 32mA. The difference of current between with and without security suite, from one aspect, demonstrates the effectiveness of security suite, which happens to be an loophole leveraged by the adversary.

TABLE 2: Node working current.

Working mode	Average current
Idle	11mA
Pure data processing	18mA
Receiving and processing unsecured packets	32mA
Receiving and processing secured packets	40mA

7.1 Single-hop Scenario

A simple single-hop network consisting of an attack node and a pair of transmitter and receiver/victim nodes is deployed in the experiments. The transmitter and victim nodes, separated by a distance of 0.5m, operate in a low duty-cycling mode. Specifically, they wake up (by turning on their radios) and stay active for roughly

5ms every 50ms. Due to clock drift and the consequent problem of asynchronous communication rendezvous in duty-cycling networks, it is possible that the receiver does not receive anything from the transmitter in one active period. The *ghost* attack node is placed close to both the transmitter and victim nodes. It broadcasts 25 bogus packets per second, where each packet has a payload size of 60 Bytes.

We measure the lifetime of the victim node which is powered by a 70-mAh Li-Ion rechargeable battery. We conduct three groups of independent experiments with three batteries. In order to have the same initial energy, the batteries are charged for a same period of time after they have been completely depleted. We assume the *ghost* has unlimited resource; thus, it is powered by a D.C. stabilized power supply. The transmitter node's power supply is a normal AA battery which has much higher initial energy than that of the victim node.

TABLE 3: Node lifetime in single-hop scenario. Unit is minute.

			Battery			Average	
			#1	#2	#3	Lifetime	Percent
Experimental	No attack		86	90	100	92	100%
	Under <i>ghost</i> attack	011	56	48	65	56	60.9%
		100	62	57	71	63	68.5%
		111	52	50	60	54	58.7%
Mapped	No attack		3858	4285	4458	4200	100%
	Under <i>ghost</i> attack	011	283	215	329	275	6.6%
		100	333	275	379	329	7.8%
		111	252	227	291	257	6.1%

The experiment results are shown in Table 3 which clearly justifies the effectiveness of *ghost* attack since the node lifetime is obviously shortened under attack. Particularly, with the AES-CCM-128 security suite, the node lifetime is significantly reduced by 39.5%, 44.4% and 40.0% with respect to the 3 batteries used, respectively. Moreover, the percentages of lifetime reduction also climbs as the security level increases.

For ease of implementation, the experiments use different settings compared to our simulations in Section 6.2, for which reason the results in Table 3 are different from those shown in Fig. 4. Beside using different battery models and different initial battery power, there are three other major differences between experiments and simulations: 1) A transmitter is introduced to inject packets to the victim node in the experiments. 2) The duty cycle of the victim node is 10% in the experiments, larger than that in the simulations (which is 1%). 3) In our simulations, once the victim node enters sleep period, it turns off its radio and switches its CPU to power-save mode. However, in our experiments, the victim node only switches off its radio (while keeping its CPU in idle mode) during its sleep period.

Based on the measured node working current shown in Table 2, the experiment results in Table 3 are mapped to those shown in Table 3 by removing the transmitter,

using 1% as the victim node's duty cycle and switching its CPU to power-save mode once it begins sleeping. We can observe that, under the *ghost* attack, the node lifetime is remarkably reduced to 6%~8% of the lifetime when the attacker is not present. The percentages of lifetime reduction are very close to those observed in the simulations shown in Fig. 4. Therefore, our experiments both confirm the significance of the *ghost* attack and validate our methodology of simulations.

7.2 Multi-hop Scenario

We construct a multi-hop network whose topology is the same as that displayed in Fig. 1. All the legitimate nodes turn on the radio for 5ms in every 50ms and transmit the packets to the sink node through the fixed path shown in the figure. The ghost attack strategy is implemented in a MicaZ node, which uses the same strategy as in the above single-node scenario to attack Node 2. Each node uses the AES-CCM-128 as its security suite. The experiment results of four representative nodes (Node 1, 2, 3 and 5) are shown in Table 4. We can observe that Node 2, the target of the *ghost*, suffers from the fastest speed of energy draining. Its lifetime is shortened as much as 31.6%, which basically approaches the performance of the victim node in our single-hop scenario (as compared to the experimental results in Table 3). Since Node 1 and 3 also receive the bogus packets, their lifetime is decreased by 15.8% and 16.2%, respectively. Since the traffic of Node 1,2 and 3 are suppressed by the *ghost*, the throughput of Node 5 increases a little bit, which causes 9.5% reduction of its lifetime. This trend of Node 5 is also demonstrated in Fig. 2 based on our previous analytical model.

TABLE 4: Node lifetime in the multi-hop scenario.

		Battery Set			Average	
		#1	#2	#3	Lifetime	Percent
No at- tack	N_1	85 min	80 min	81 min	82 min	100%
	N_2	81 min	78 min	79 min	79 min	100%
	N_3	78 min	71 min	72 min	74 min	100%
	N_5	88 min	84 min	80 min	84 min	100%
Under <i>ghost</i> attack	N_1	74 min	64 min	68 min	69 min	84.1%
	N_2	53 min	52 min	59 min	54 min	68.4%
	N_3	60 min	62 min	64 min	62 min	83.8%
	N_5	77 min	73 min	77 min	76 min	90.5%

8 CONCLUSION

In this paper, we present a novel and severe attack termed as *ghost* in which the malicious one transmits a number of bogus messages to lure the receiving victim device to do the superfluous security-related computations, leading to battery depletion. Our simulation results manifest that by launching this attack, an attacker could easily reduce the lifetime of ZigBee devices from years to days. We also demonstrate that the *ghost* attack can further trigger severe DoS and post-depletion

attacks. We propose several recommendations on how to withstand the *ghost* and other related attacks in ZigBee networks. Extensive simulations are provided to show the impact of the *ghost* attack and the performance of the proposed recommendations. To further validate the effectiveness of *ghost* attack, physical experiments were conducted on ZigBee nodes and interestingly, our results show that the lifetime of nodes are significantly impacted with this attack. We believe that the presented work will aid the researchers to improve the security of ZigBee networks further.

REFERENCES

- [1] S. Poslad, *Ubiquitous computing: smart devices, environments and interactions*. Wiley, 2011.
- [2] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.
- [3] Y. He, W. Zhu, and L. Guan, "Optimal resource allocation for pervasive health monitoring systems with body sensor networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 11, pp. 1558–1575, 2011.
- [4] N. A. Khan, N. Javaid, Z. A. Khan, M. Jaffar, U. Rafiq, and A. Bibi, "Ubiquitous healthcare in wireless body area networks," in *IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012, pp. 1960–1967.
- [5] K. Chebrolu and A. Dhekne, "Esense: energy sensing-based cross-technology communication," *IEEE Trans. Mobile Computing*, vol. 12, no. 11, pp. 2303–2316, 2013.
- [6] C. Neuman and K. Tan, "Mediating cyber and physical threat propagation in secure smart grid architectures," in *IEEE Int. Conf. Smart Grid Communications (SmartGridComm)*, 2011, pp. 238–243.
- [7] IEEE Computer Society, "Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)," IEEE Standard 802.15.4, 2006.
- [8] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *Proc. ACM workshop on Wireless security*, 2004, pp. 32–42.
- [9] J. Zheng, M. J. Lee, and M. Anshel, "Toward secure low rate wireless personal area networks," *IEEE Trans. Mobile Computing*, vol. 5, no. 10, pp. 1361–1373, 2006.
- [10] KillerBee, <http://code.google.com/p/killerbee>.
- [11] Y. Xiao, H. Chen, B. Sun, R. Wang, and S. Sethi, "MAC security and security overhead analysis in the IEEE 802.15.4 wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2006, 2006.
- [12] M. Doomun, K. Soyjaudah, and D. Bundhoo, "Energy consumption and computational analysis of Rijndael-AES," in *IEEE/IFIP Int. Conf. in Central Asia on Internet*, 2007, pp. 1–6.
- [13] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal jamming attack strategies and network defense policies in wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 8, pp. 1119–1133, 2010.
- [14] J. Tang, Y. Cheng, and W. Zhuang, "Real-time misbehavior detection in IEEE 802.11-based wireless networks: an analytical approach," *IEEE Trans. Mobile Computing*, vol. 13, no. 1, pp. 146–158, 2014.
- [15] A. H. Lashkari, M. M. S. Danesh, and B. Samadi, "A survey on wireless security protocols (WEP, WPA and WPA2/802.11i)," in *IEEE Int. Conf. Computer Science and Information Technology*, 2009, pp. 48–52.
- [16] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proc. IEEE INFOCOM*, vol. 3, 2002, pp. 1530–1539.
- [17] Q. Dong, D. Liu, and P. Ning, "Providing dos resistance for signature-based broadcast authentication in sensor networks," *ACM Trans. Embedded Computing Systems*, vol. 12, no. 3, p. 73, 2013.
- [18] A. Mpitiopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009.
- [19] M. Brownfield, Y. Gupta, and N. Davis, "Wireless sensor network denial of sleep attack," in *Annual IEEE SMC Information Assurance Workshop*, 2005, pp. 356–364.
- [20] X. Hei, X. Du, J. Wu, and F. Hu, "Defending resource depletion attacks on implantable medical devices," in *IEEE GLOBECOM*, 2010, pp. 1–5.
- [21] E. Y. Vasserman and N. Hopper, "Vampire attacks: draining life from wireless ad hoc sensor networks," *IEEE Trans. Mobile Computing*, vol. 12, no. 2, pp. 318–332, 2013.
- [22] J. Zheng and M. Lee, "Will IEEE 802.15.4 make ubiquitous networking a reality? a discussion on a potential low power, low bit rate standard," *IEEE Communications Magazine*, vol. 42, no. 6, pp. 140–146, 2004.
- [23] D. Shila, Y. Cheng, and T. Anjali, "Mitigating selective forwarding attacks with a channel-aware approach in WMNs," *IEEE Trans. Wireless Communications*, vol. 9, no. 5, pp. 1661–1675, 2010.
- [24] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.
- [25] X. Ling, Y. Cheng, J. W. Mark, and X. Shen, "A renewal theory based analytical model for the contention access period of IEEE 802.15.4 MAC," *IEEE Trans. Wireless Communications*, vol. 7, no. 6, pp. 2340–2349, 2008.
- [26] NS-3, <http://www.nsnam.org>.
- [27] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. 2nd ACM Int. Conf. embedded networked sensor systems (Sensys)*, 2004, pp. 188–200.
- [28] C. Shepherd, "Design of primary and secondary cells ii. an equation describing battery discharge," *Journal of the Electrochemical Society*, vol. 112, no. 7, pp. 657–664, 1965.
- [29] O. Tremblay, L. Dessaint, and A. Dekkiche, "A generic battery model for the dynamic simulation of hybrid electric vehicles," in *IEEE Vehicle Power and Propulsion Conference*, 2007, pp. 284–289.
- [30] S. Liu, K. Fan, and P. Sinha, "CMAC: an energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Trans. Sensor Networks*, vol. 5, no. 4, p. 29, 2009.
- [31] A. Tsertou and D. Laurenson, "Revisiting the hidden terminal problem in a CSMA/CA wireless network," *IEEE Trans. Mobile Computing*, vol. 7, no. 7, pp. 817–831, 2008.